A 128∼2048/1536 POINT FFT HARDWARE IMPLEMENTATION WITH OUTPUT PRUNING

Tuba Ayhan[1], *Wim Dehaene*[1,2], *Marian Verhelst*[1]

[1]ESAT-MICAS, KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
[2]IMEC, Kapeldreef 75, B-3001 Leuven, Belgium
{tuba.ayhan, wim.dehaene, marian.verhelst}@esat.kuleuven.be

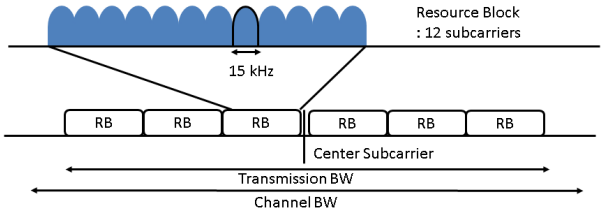
ABSTRACT

In this work, an FFT architecture supporting variable FFT sizes, 128∼2048/1536, is proposed. This implementation is a combination of a 2^p point Common Factor FFT and a 3 point DFT. Various FFT output pruning techniques for this architecture are discussed in terms of memory and control logic overhead. It is shown that the used Prime Factor FFT as an FFT in the 1536 point FFT is able to increase throughput by exploiting single tone pruning with low control logic overhead. The proposed FFT processor is implemented on a Xilinx Virtex 5 FPGA. It occupies only 3148 LUTs and 612 kb memory in FGPA and calculates 1536 point FFT less than 3092 clock cycles with output pruned settings.

Index Terms— FFT Pruning, FPGA Implementation, LTE, Variable size FFT, Prime Factor FFT

1. INTRODUCTION

Signal processing applications are subjected to increase energy efficiency and processing throughput. The Fast Fourier Transform (FFT) is one of the most essential algorithms of signal processing. Numerous architectures for FFT implementations are proposed to achieve either the highest throughput, minimum area, energy efficiency, resource optimization, minimum memory usage or flexibility. So far in many applications and standards, the FFT window size is selected as 2^N because the Common Factor FFT (CF-FFT) which is also known as Cooley-Tukey FFT, provides an efficient implementation for an input and output size of 2^N where N is an integer. However, with the enhancements in wide-band technologies, research has shifted towards FFT architectures optimized for variable length and sparse FFT outputs. Applications in communication technologies, channel estimation [1] and localization systems [2] tend to use wider bands which contain spread spectral information among the channel bandwidth. The required FFT window size to process this data is not necessarily of size 2^N. Mixed-radix architectures [3,4] are proposed as an alternative to radix-2 and radix-4 CF-FFT architectures. The emerging communication standard, LTE (Long Term Evolution) requires 128-2048/1536 size FFT, providing a flexible spectrum support from 1.4 up to 20 MHz [5]. In LTE systems, information is transmitted in resource blocks each containing 12 subcarriers. The number of resource blocks can be increased and distributed in both time and frequency when needed. Using OFDMA (Orthogonal Frequency Division Multiple Access), the occupied subcarriers will hence vary over time, according to the current transmission bandwidth and the allocated resource blocks. In order to save energy, throughput and area through the dynamic bandwidth and resource block allocation principles of LTE, two new implementation challenges arise in the research community:



Channel Bandwidth (MHz)	1.4	2.5	5	10	15	20
Resource Blocks	6	15	25	50	75	100
Max subcarrier	72	180	300	600	900	1200
FFT Size	128	256	512	1024	1536	2048

Fig. 1. LTE resource blocks, transmission and channel bandwidth

1. Online programmable 128∼2048/1536 point FFT processing: Considering the channel bandwidth and FFT sizes for each band given in Figure 1, 128∼2048/1536 point FFT implementations are proposed in [6–8].
2. Efficiently pruning FFT output: Not all FFT outputs are useful, because of the resource block allocation. Pruning the unnecessary operations in the FFT reduces energy consumption and increases throughput.

There are numerous FFT pruning techniques presented in literature but only few of them can be implemented on hardware, because the gain in pruning does not compensate for the cost of the pruning control unit. Therefore, FFT pruning is used only when the signal is very sparse in the frequency

This project is funded by Flemish Agency for Innovation by Science and Technology (IWT) as SBO project OmniTrack.

domain [9], or the frequency bins to be pruned are fixed.

None of the mentioned techniques fulfill the requirements of an efficient FFT processor that can work for variable FFT sizes and different output pruning matrices. In this work, we present an FFT processor to be used in LTE systems providing the desired flexibility. The FFT processor, which supports 128∼2048/1536 point FFT as well as output pruning, accomplishes both transmission bandwidth and resource block allocation flexibility beneficial to LTE systems. In this work, we are using a core block of a 128 point FFT with controllable twiddle factors to support 128∼2048 point FFT. We are combining PF-FFT (Prime Factor FFT) with CF-FFT to support $3 \cdot 2^9 = 1536$ point FFT. Moreover, the control unit that controls the overall operation is able to dynamically exclude blocks for processing for selective output pruning.

This paper is organized as follows: first, a theoretical overview on CF-FFT, PF-FFT and FFT pruning is given in Section 2. Then the pruned FFT architecture combining CF-FFT and PF-FFT is given in Section 3. The hardware implementation of this architecture is presented and compared with two implementations that tackle similar challenges in Section 4. Finally, the paper is concluded in Section 5.

2. OVERVIEW ON FFT

A DFT (Discrete Fourier Transform) can be accelarated by reducing the complexity of the operation given as

$$X_k = \sum_{n=0}^{N-1} x_n \cdot exp\left(-\frac{2\pi i}{N} nk\right) \quad k = 0, ..., N-1. \quad (1)$$

FFT algorithms calculate the outputs of the DFT much faster. Further decreasing the computation time of the FFT is possible with pruning, under certain conditions. Overviews of two FFT algorithms and FFT pruning are given in this section.

2.1. Common Factor FFT

CF-FFT reduces the computation complexity when the FFT size is r^p where p and r are integers. r is the radix base of the FFT. Typically, the radix base is 2 and 4; which are sufficient for many conventional FFT operations. Alternatively, radix 3 and 5 architectures are also proposed for LTE systems [3].

CF-FFT for an FFT size of 2^p where $p_1 < p < p_2$ can be implemented using one 2^{p_1} CF-FFT block with controllable twiddle factors and a control unit.

2.2. Prime Factor FFT

PF-FFT is based on factorization of the FFT size into mutually prime factors. Although, the number of factors is not limited, in this work an FFT of size N will be divided into two factors, N_1 and N_2, given as

$$N = N_1 . N_2 \quad (2)$$

provided that N_1 and N_2 are mutually prime. PF-FFT re-indexes n and k in (1). n and k becomes

$$n = n_1 N_2 + n_2 N_1 \bmod N, \quad (3)$$

$$k = k_1 \text{minv}(N_2, N_1) N_2 + k_2 \text{minv}(N_1, N_2) N_1 \bmod N, \quad (4)$$

where operation $\text{minv}(x, y)$ denotes modular multiplicative inverse of x modulo y. After simplifications, the re-indexing results in [10]

$$X_k = \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} x_{n_1 N_2 n_2 N_1} \cdot exp\left(-\frac{2\pi i}{N_2} n_2 k_2\right) \right) exp\left(-\frac{2\pi i}{N_1} n_1 k_1\right). \quad (5)$$

By (5), an N_2 point DFT operation inside an N_1 point DFT is obtained. In total, there are N_1 DFT operations of N_2 points, and one N_1 point DFT. The N_1 point DFT operation is repeated for N_2 times, producing $N_1 \times N_2 = N$ outputs as desired.

For small numbers ($M < 8$), $2^M - 1$ prime factor factorization requires less resources than 2^M point Cooley Tukey based FFT [10]. However, prime factor factorization based FFT algorithms are currently implemented in software, no hardware implementations exist.

2.3. FFT Pruning

Wider bandwidths are under consideration to reach higher data rates but in order to maintain energy efficiency, the signal is sparsely distributed in the spectrum, upon application needed. In other words, not the entire available spectrum but only the selected parts of it contain necessary information, so only a subset of output bins are needed. Therefore, calculating the spectral information of each frequency bin is an over-design. In order to eliminate this burden, researchers proposed sparse FFT [9] and pruned FFT techniques for cognitive radio and spectrum sensing [11–13]. By pruning the outputs, the butterfly branches are pruned, so the number of multiplications and additions is decreased. To eliminate operations for unnecessary outputs conditional statements are used. These conditional statements increases computation time and not favored for hardware implementation [14, 15]. To reduce this burden, reconfigurable FPGA [16] and instruction level parallel architectures [13] are proposed. However, all mentioned pruning techniques are based on 2^N point FFT.

3. ARCHITECTURE

The FFT architecture of this work is a combination of the widely used CF-FFT with a DFT. This architecture for the 1536 point FFT can be seen in Figure 2. Left side of the figure is the CF-FFT part, the outputs of this part are used by the DFT processors on the right side.

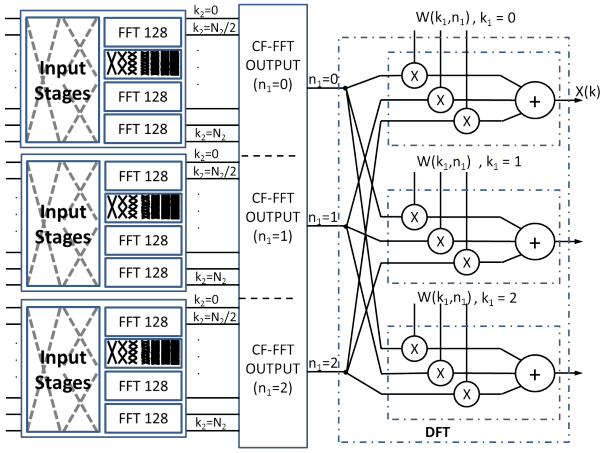


Fig. 2. PF-FFT to calculate 1536 point FFT

3.1. 128∼2048 Point FFT and Block Pruning

For the 128∼2048 point FFT 7∼11 FFT stages are needed. A 128 point FFT has 7 stages which comprises the 7 stages close to the FFT output for the 256∼2048 point FFT. For a 2^M point FFT, $2^M/128$ times the 128 point FFTs and $M-7$ input stages are needed. For example, as given in the Figure 2, 2 extra stages are needed to implement a 512 point FFT. After the input data is processed by these input stages, a 128 point FFT is run four times. The twiddle factors of both the input stages and the 128 point FFT are programmable according to the CF-FFT size.

With this architecture, block-wise pruning of the FFT output is feasible. After the input stages are processed, one or more 128 point FFTs can be pruned depending on the desired output information. Block pruning is possible when the number of outputs to be pruned is greater than 128 and they are contiguous in natural or bit-reversed order. As expected, this is a very rare situation; only a small portion of all possible subcarrier combinations are valid for pruning. As the number of the informative subcarriers decrease, the signal becomes more sparse and the probability to meet a subcarrier configuration that is valid for block pruning increases. Figure 3 shows the probability of valid block pruning versus sparsity. When the sparsity of the signal is less than 2%, only 10% to 45% of the possible subcarrier combinations are suitable for block pruning. Therefore, the advantage of block pruning is negligible.

3.2. 1536 Point FFT and Single Tone Pruning

To build a 1536 point FFT, the prime factors of the PF-FFT are chosen as $N_1 = 3$ and $N_2 = 2^9$. As explained in Section 2.2, three 512 point CT-FFT operations are followed by a 3 point DFT. This block operation can significantly benefit from single tone output pruning, as we will illustrate below.

An example of output pruning in the $2^M \times 3$ point FFT, for $M = 4$, is given in Figure 4. The output indexes, k_1

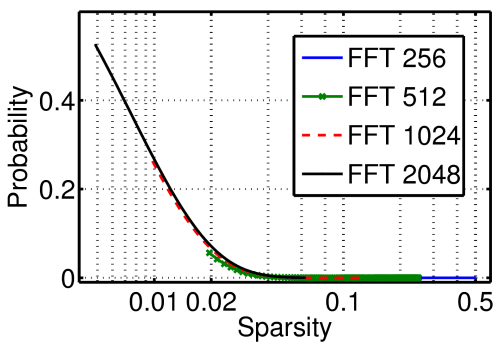


Fig. 3. Probability of valid block pruning versus sparsity in the frequency domain

k2 \ k1	0	1	2
0	0	16	32
1	33	1	17
2	18	34	2
3	3	19	35
4	36	4	20
5	21	37	5
6	6	22	38
7	39	7	23
8	24	40	8
9	9	25	41
10	42	10	26
11	27	43	11
12	12	28	44
13	45	13	29
14	30	46	14
15	15	31	47

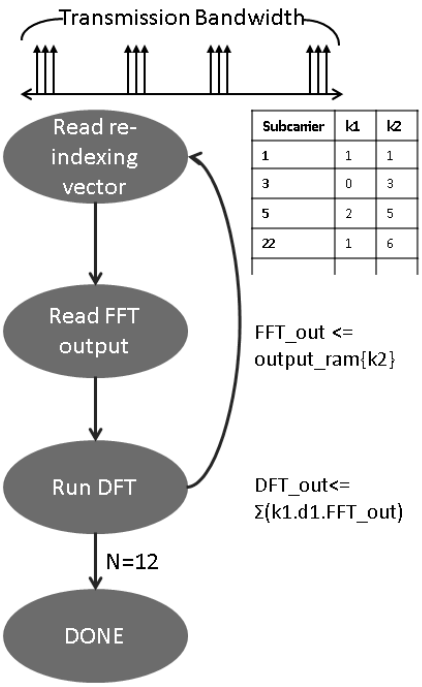


Fig. 4. Output pruning example with PF-FFT

and k_2, are the output indexes of the DFT and the CF-FFT, respectively. In the first step, all CF-FFT outputs are calculated. Therefore, all the outputs for $k_2 = [1, 2^M]$ are ready before the DFT operation.

Single tone pruning is implemented by means of output tone selection by the DFT control unit, therefore it is viable for the 1536 point FFT. When pruning is enabled, the DFT works only for the necessary combinations of CF-FFT, k_2, and DFT, k_1. For example, to calculate frequency bin 16, $X(k = 16)$, the DFT reads the CF-FFT outputs at $k_2 = 0$. Inside the DFT unit, the multiply-add block which uses the coefficients (W) with $k_1 = 1$ is run. This way, the outputs which are not contiguous can be prunned.

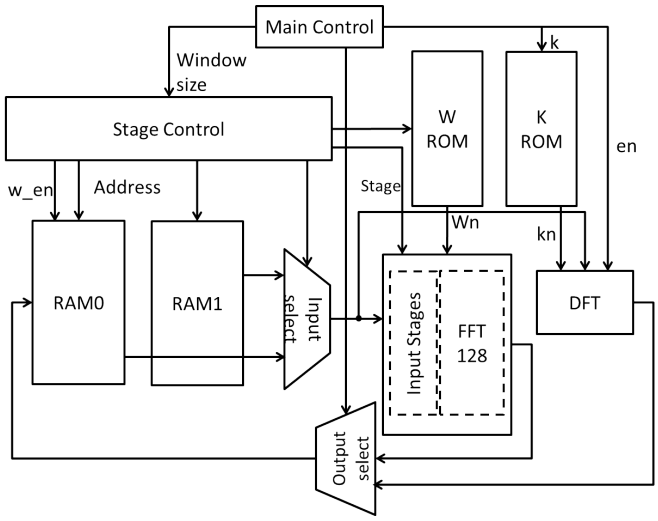


Fig. 5. Programmable 128∼2048/1536 point FFT

	128∼2048 /1536 point with pruning	**128∼2048 point**	**128∼512 point**
Occupied slices	787	728	581
DSP48Es	30	14	14
Total Memory (kb)	612	594	252
RAM Blocks	22	22	9

Table 1. Resources for variable length FFT

4. HARDWARE IMPLEMENTATION

Radix-2 DIT (Decimation in Time) FFT is implemented using a 128 point FFT core, a DFT unit, control units and RAMs as given in Figure 5. The operation runs as follows: the main control unit initializes CF-FFT size for the FFT control, DFT unit and the interaction between them. The size of CF-FFT means how many input stages will be used and how many times the 128 point FFT core will operate.

The input stages and twiddle factors are controlled by Stage Control. RAM0 and RAM1 become input and output RAM at each stage in alternating turn.

For the 1536 point FFT, after the 512 point CF-FFT is performed for 3 times, the output of the CF-FFT is read from RAM1 and the DFT output is written to RAM0. When the pruning is enabled, the CF-FFT output to be processed is selected by Main Control via Stage Control. The exponential factor for the DFT unit is also selected by Main Control. Implementation is done on a Xilinx Virtex-5 (XC5VFX130T-2FFG1738CES) FPGA. The resource usage is given in Table 1.

As given in Section 2.2, re-indexing uses modular arithmetic and is not straightforward to generate inside an FPGA.

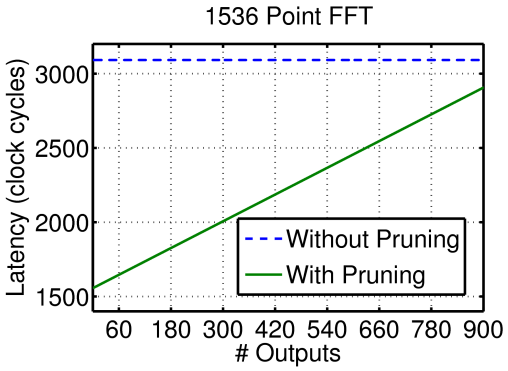


Fig. 6. Advance in latency with output pruned 1536 point PF-FFT

Therefore, writing pre-calculated indexes into a look-up table is preferred over generating the indexes on the FPGA. The DFT and FFT units remain as they are, but a re-indexing vector is necessary for pruning. The re-indexing vector which points the k_1 and k_2 indexes to be processed should be stored in a memory. k_1 and k_2 addresses are 2 and 9 bits long respectively. The total memory for re-indexing is 9900 bits for 900 subcarriers. That memory is the overhead of pruning. Using this pruning method, the latency can be decreased from 3092 clock cycles down to 1646 cycles when 5 LTE resource blocks (60 tones) are occupied. The improvement in latency for 36 to 900 subcarriers for a 1536-point FFT is given in Figure 6.

4.1. Comparison

To the best of our knowledge, none of previously proposed FFT architectures for LTE supports variable window length as well as output pruning at the same time. So, the architecture is compared with two hardware efficient and high throughput FFT architectures for LTE systems given in [6] which is synthesized on 130 nm ASIC technology and [8] which is implemented on Virtex-5 FPGA. Both of these designs employ radix-3 support to compute 1536 point FFT.

These proposed architectures are compared by resource utilization and computation time in clock cycles in Table 2. Using PF-FFT instead of a mixed-radix architecture, leads to a significant increase in throughput in typical LTE workloads because it enables output pruning.

5. CONCLUSION

In this work, an FFT architecture supporting both variable window sizes and output pruning is proposed. The FFT supports 128∼2048/1536 point FFT, as required by LTE systems. Other FFT architectures proposed for LTE systems do not enable output pruning. Using PF-FFT, our architecture facilitates output pruning, decreasing the computation time by

	This work	[6]	[8]
LUTs (Virtex-5)	3148	NA	23807
Total Memory (kb)	612	NA	224
Area (kgates)	NA	109	NA
Computation clk cycles (2048 pnt)	8195	12345	3072
Computation clk cycles (1536 pnt, 60-900 sub-carriers)	1646-2906	9324	4224

Table 2. Comparison

half. The throughput can be doubled by means of output pruning, with an overhead of a re-indexing memory of maximum 1.2 KB. Therefore, the architecture implemented on a Virtex-5 FGPA increases throughput while maintaining the area.

REFERENCES

[1] J. Lofgren, O. Edfors, and P. Nilsson, "Improved Matching Pursuit Algorithm and Architecture for LTE Channel Estimation," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, May 2011, pp. 466–469.

[2] T. Ayhan, T. Redant, M. Verhelst, and W. Dehaene, "Towards a Fast and Hardware Efficient Sub-mm Precision Ranging System," in *Proceedings of IEEE Workshop on Signal Processing Systems (SiPS)*, 2012.

[3] J. Lofgren and P. Nilsson, "On Hardware Implementation of Radix 3 and Radix 5 FFT Kernels for LTE Systems," in *NORCHIP, 2011*, Nov 2011, pp. 1–4.

[4] Y. Suzuki, Toshio Sone, and K. Kido, "A new FFT algorithm of radix 3,6, and 12," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 2, pp. 380–383, Apr 1986.

[5] A. Ghosh and R. Ratasuk, *Essentials of LTE and LTE-A*, Cambridge University Press, 2011.

[6] T. Patyk, D. Guevorkian, T. Pitkanen, P. Jaaskelainen, and J. Takala, "Low-power application-specific FFT processor for LTE applications," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*, July 2013, pp. 28–32.

[7] Yuan-Chu Yu and Yuan-Tse Yu, "Design of a High Efficiency Reconfigurable Pipeline Processor on Next Generation Portable Device," in *Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE), 2013 IEEE*, Aug 2013, pp. 42–47.

[8] J. Chen, J. Hu, and Li S., "High Throughput and Hardware Efficient FFT Architecture for LTE Application," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, April 2012, pp. 826–831.

[9] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi, "Bigband: Ghz-wide sensing and decoding on commodity radios," 2013.

[10] R. Stasinski, "Prime factor FFT for modern computers," in *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on*, April 2012, pp. 346–349.

[11] Wen-Bin Chien, Chih-Kai Yang, and Yuan-Hao Huang, "Energy-saving cooperative spectrum sensing processor for cognitive radio system," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 58, no. 4, pp. 711–723, April 2011.

[12] Q. Zhang, A.B.J. Kokkeler, and G. J M Smit, "A reconfigurable radio architecture for cognitive radio in emergency networks," in *Wireless Technology, 2006. The 9th European Conference on*, Sept 2006, pp. 35–38.

[13] Min Li, D. Novo, B. Bougard, T. Carlson, L. Van der Perre, and F. Catthoor, "Generic Multiphase Software Pipelined Partial FFT on Instruction Level Parallel Architectures," *Signal Processing, IEEE Transactions on*, vol. 57, no. 4, pp. 1604–1615, 2009.

[14] H.V. Sorensen and C.S. Burrus, "Efficient computation of the DFT with only a subset of input or output points," *Signal Processing, IEEE Transactions on*, vol. 41, no. 3, pp. 1184–1200, 1993.

[15] R.G. Alves, P. L. Osorio, and M.N.S. Swamy, "General FFT pruning algorithm," in *Circuits and Systems, 2000. Proceedings of the 43rd IEEE Midwest Symposium on*, 2000, vol. 3, pp. 1192–1195 vol.3.

[16] C. Vennila, C.T.K. Palaniappan, K.V. Krishna, G. Lakshminarayanan, and Seok-Bum Ko, "Dynamic partial reconfigurable FFT/IFFT pruning for OFDM based Cognitive radio," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, May 2012, pp. 33–36.